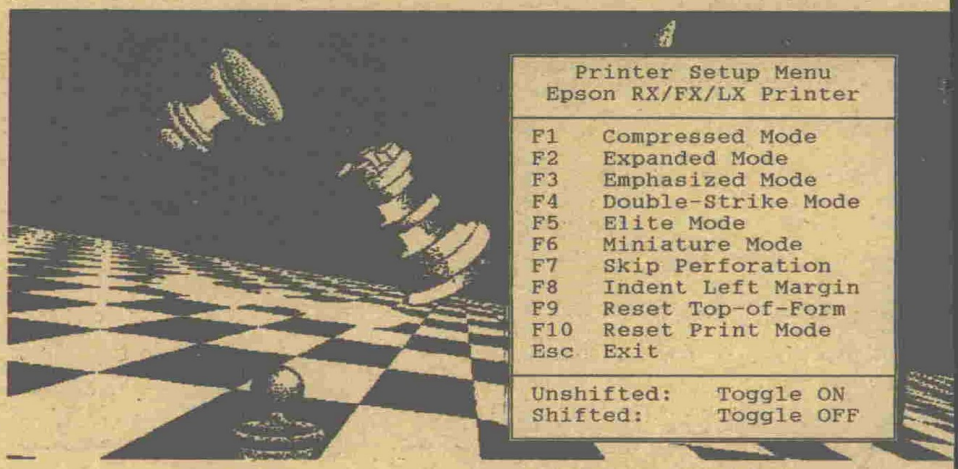
ซอฟต์แวร์
ชจร โรจนเมธินทร์

โปรแกรมควบคุม เครื่องพิมพ์

เมื่อเรากำลังใช้งานอยู่ เราสามารถคอนโทรลเครื่องพิมพ์ โดยไม่ต้องออกมาสู่ดอส ช่วยให้งานเสร็จเร็วขึ้น



ท่านผู้อ่านหลายคนคงเคยส่งคอนโทรลโค้ดไปให้กับเครื่องพิมพ์คงยุ่งยากพอดูถ้าเราต้องเข้าหรือออกจากโปรแกรมที่เรากำลังใช้งานอยู่เพื่อออกมาที่ดอสพรอมต์ในการรันโปรแกรมที่จะส่งคอนโทรลโค้ดไปที่เครื่องพิมพ์ ดังนั้นคงจะวิเศษไม่น้อยถ้าคุณสามารถส่งคอนโทรลโค้ดนั้นเมื่อใดก็ได้ตามต้องการ โดยไม่ต้องเข้าหรือออกจากโปรแกรมบ่อย ๆ SP.COM จะทำหน้าที่นี้ให้คุณได้ เพียงแต่คุณกดคีย์ Ctrl+Right Shift key พร้อมกันก็จะมีเมนูปรากฏขึ้นบนจอภาพ และรอรับคำสั่งจากคุณที่จะส่งคอนโทรลโค้ดไปควบคุมเครื่องพิมพ์ตามต้องการ SP.COM ใช้ได้กับเครื่องพิมพ์เอ็ปซอนรุ่น RX, FX, LX และยังสามารถแก้ไขให้ใช้กับยี่ห้ออื่นได้ โดยเปลี่ยนรหัสของคอนโทรลโค้ด SP.COM จะซ่อนตัวอยู่บนหน่วยความจำ กินเนื้อที่เพียงประมาณ 3 กิโลไบต์ แต่จะช่วยให้การทำงานมีประสิทธิภาพมากขึ้น

การใช้งาน SP.COM

หลังจากที่โหลดโปรแกรมลงบนหน่วยความจำแล้ว SP.COM จะฝังตัวอยู่บนหน่วยความจำ ซึ่งคุณสามารถเรียก SP.COM มาทำงาน โดยผ่านคีย์บอร์ดสำหรับ SP.COM จะเรียกโดยการกดคีย์ Ctrl+Right Shift key พร้อมกัน จะปรากฏเมนูขึ้นบนจอภาพแล้วโปรแกรมจะรอรับคำสั่ง ซึ่งคุณสามารถส่งคอนโทรลโค้ดต่าง ๆ ที่มีให้เลือกบนเมนูไปเซตเครื่องพิมพ์ได้ โดยกดฟังก์ชันคีย์ F1-F10 และสามารถยกเลิกการเซตเครื่องพิมพ์นั้นโดยกด Shift key + ฟังก์ชันคีย์ (F1-F10) เช่น ถ้าคุณต้องการเซตให้เครื่องพิมพ์ พิมพ์อักษรแบบตัวเล็ก (Compressed mode) ก็ให้กด F1 แต่ถ้าต้องการยกเลิกการพิมพ์ตัวอักษรแบบตัวเล็ก ก็ให้กด Shift+F1 เป็นต้น จากนั้นเมื่อต้องการเลิกการใช้ SP.COM ก็ให้กดคีย์ Esc คุณก็จะกลับไปยังการทำงานที่จุดเดิมก่อนหน้าที่จะเรียก SP.COM มาใช้

สำหรับกรณีที่เครื่องพิมพ์ยังไม่อยู่ในสภาพออนไลน์ หรือ คุณกดคีย์ที่นอกเหนือจากที่กำหนด SP.COM จะส่งเสียงเตือนให้ทราบว่า มีความผิดพลาดเกิดขึ้น

การทำงานของ SP.COM

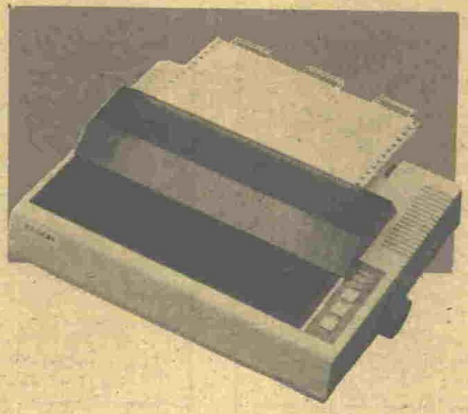
เมื่อคุณรัน SP.COM ครั้งแรก โปรแกรมจะตรวจสอบชนิดของจอภาพเพื่อเลือกใช้แอตทริบิวต์ได้ถูกต้อง จากนั้นก็จะกำหนดคำลงบนหน่วยความจำบริเวณ WINDOW_BYTES แต่เราจองไว้โดยใช้เลเบลชื่อ WINDOW_TEXT เป็นเวิร์ดทั้งหมด 532 เวิร์ด (28×5) ก็เพื่อความสะดวกในการเขียนลงบนหน่วยความจำของจอภาพด้วยคำสั่ง MOVSW ให้เก็บตัวอักษรและแอตทริบิวต์ของเมนู ซึ่งจะใช้เขียน

โดยตรงลงบนหน่วยความจำของจอภาพ และเปลี่ยนแอดเดรสของอินเตอร์รัพต์เวคเตอร์ของคีย์บอร์ดให้ชี้มาที่ NEW9H และจบโปรแกรมด้วยอินเตอร์รัพต์ 27h เมื่อมีการกดคีย์บอร์ด จะทำให้มีการเรียกใช้อินเตอร์รัพต์ 9h นั่นก็คือ SP.COM จะเริ่มต้นทำงานโดยการเรียกอินเตอร์รัพต์ 9h ของเดิมก่อน จากนั้นก็จะตรวจสอบสถานะของ Shift key โดยใช้อินเตอร์รัพต์ 16h ว่ามีการกดคีย์ Ctrl+Right Shift key หรือไม่ ถ้าไม่มีก็จะออกจากโปรแกรมโดย IRET แต่ถ้ามีการกดก็จะตรวจสอบให้แน่นอนว่า ขณะนั้นเมนูของ SP.COM ยังไม่ถูกเขียนบนจอภาพจากนั้นจะเก็บอักขระและแอตทริบิวต์ บริเวณซึ่งเราจะเขียนเมนูทับลงไปไว้ที่ WINDOW_BUFFER เพื่อใช้เขียนกลับคืนเมื่อออกจาก SP.COM เมื่อเขียนเมนูลงบนจอภาพแล้ว โปรแกรมก็จะรอรับค่าจากคีย์บอร์ด โดยอินเตอร์รัพต์ 16h ถ้ามีการกดคีย์ F1-F10, Shift-F1-Shift-F10 หรือมิฉะนั้น จะลูปรอรับค่าจากคีย์บอร์ด จนกว่าจะมีการกด Esc เพื่อออกจาก SP.COM โปรแกรมก็จะใช้ค่าสแกนโค้ดที่รับแปลงให้เป็นออฟเซ็ตแอดเดรส เพื่อใช้อ่านค่าสตริงของคอนโทรลโค้ดที่จะส่งให้เครื่องพิมพ์ (LPT1:) ทางอินเตอร์รัพต์ 17h

สำหรับสตริงของคอนโทรลโค้ดนั้น กำหนดให้มีความยาวคงที่ 16 ไบต์ เพื่อให้สะดวกในการคำนวณค่าออฟเซ็ทแอดเดรสของแต่ละสตริง สตริงของคอนโทรลโค้ดนี้จะใช้ค่า 255 เป็น terminator ของสตริง

หิมะบนจอภาพ CGA

การเขียนค่าอักษรลงบนหน่วยความจำของจอภาพ CGA โดยตรงนั้นอาจก่อให้เกิดลักษณะคล้ายหิมะบนจอภาพ แต่สามารถแก้ไขโดยเราจะต้องเขียนค่าอักษรลงบนหน่วยความจำของจอภาพช่วง Vertical retrace ในการสแกนของจอคือ จะต้องรอให้ Horizontal retrace ผ่านไปก่อนจึงเขียนค่า ซึ่งเราจะทราบได้โดยการตรวจสอบ Vertical Sync bit (บิตที่ 3) ของ CGA Status register ทางพอร์ตหมายเลข 3DAH โดยจะรอให้ค่าของบิตที่ 3 เปลี่ยนจาก 0 เป็น 1 (คือ ขณะ Vertical retrace) จึงจะเขียนค่าอักษรลงไปได้ แต่วิธีนี้จะทำงานช้า ดังนั้น SP.COM จะใช้วิธี Disable จอภาพ โดยส่งค่า 25h ให้ Mode control register ทางพอร์ตหมายเลข 3D8h ในขณะที่จอภาพกำลังอยู่ในช่วง Vertical retrace จากนั้นเราก็สามารถเขียนค่าต่าง ๆ ลงบนหน่วยความจำของจอ CGA ได้อย่างอิสระโดยไม่มีหิมะเกิดขึ้น และเมื่อเขียนเสร็จก็จะส่งค่า 2Dh, 29h (ตามแต่ชนิดของจอว่าเป็น 80×25 COLOR หรือ 80×25 B/W) ให้ Mode control register อีกครั้ง เพื่อ Enable จอภาพให้ทำงานต่อ วิธีนี้จะทำงานได้เร็วกว่าวิธีแรกมาก



สตริงของคอนโทรลโค้ด

1. การพิมพ์ใน Compressed mode คือพิมพ์อักษรที่มีขนาดเล็กกว่าปกติ เราจะต้องส่งค่า 15 ให้กับเครื่องพิมพ์ และจะยกเลิก Compressed mode โดยส่งค่า 18

2. การพิมพ์ใน Expanded mode ตัวอักษรที่พิมพ์ในแบบนี้จะมีขนาดใหญ่กว่าปกติ เราจะต้องส่งสตริงคอนโทรลโค้ด ดังนี้ 27, 87, n โดยถ้าค่า n เท่ากับ 1 จะเป็นการเลือกพิมพ์ Expanded mode และค่า n เท่ากับ 0 จะยกเลิกการพิมพ์ Expanded mode

3. การพิมพ์ใน Emphasized mode จะทำให้ตัวอักษรที่พิมพ์มีความเข้มมากขึ้น เพราะอักษรแต่ละตัวถูกพิมพ์สองหน ขณะที่หัวพิมพ์วิ่งหนึ่งครั้ง โดยจะต้องส่งคอนโทรลโค้ดไปที่เครื่องพิมพ์ ดังนี้ 27, 69 และจะยกเลิกการพิมพ์ Emphasized mode โดยส่งค่า 27, 70

4. การพิมพ์ใน Double-strike mode จะพิมพ์ตัวอักษรแต่ละตัว 2 ครั้ง การพิมพ์แบบนี้จะทำให้เสียเวลามากขึ้น เพราะหัวพิมพ์ต้องพิมพ์บรรทัดละ 2 ครั้ง การเลือกพิมพ์ Double-strike mode เราจะต้องส่งค่า 27, 71 และจะยกเลิกการพิมพ์ โดยส่งค่า 27, 72 ให้เครื่องพิมพ์

5. การพิมพ์แบบ Elite คือ การพิมพ์ตัวอักษร 12 ตัว ต่อ 1 นิ้ว จะเลือกพิมพ์ตัวอักษรแบบนี้ โดยส่งค่า 27, 77 และยกเลิกให้กลับมาพิมพ์ตัวอักษรแบบ 10 ตัว ต่อ 1 นิ้ว โดยส่งค่า 27, 80

6. การพิมพ์ใน Miniature mode จะเป็นการพิมพ์ตัวอักษรขนาดเล็กมาก ซึ่งเราต้องเลือกพิมพ์แบบ Compressed mode ก่อน โดยส่งค่า 15 จากนั้นเลือกพิมพ์แบบ Superscript โดยส่งค่า 27, 83, 0 และกำหนดให้ line spacing มีค่า 6/72 นิ้ว โดยส่งค่า 27, 65, 6 สำหรับวิธียกเลิกการพิมพ์แบบ Miniature mode คือ ส่งค่า 18, 27, 84, 27, 50

7. Skip over perforation คือการไม่พิมพ์ตรงรอยปรุของกระดาษต่อเนื่อง ซึ่งทำได้โดยกำหนดขอบล่างของกระดาษ โดยส่งค่า 27, 78, 12 เพื่อกำหนดให้ขอบล่างของกระดาษอยู่เหนือรอยปรุ 12 บรรทัด และยกเลิกการกำหนดขอบล่างของกระดาษ โดยส่งค่า 27, 79

8. กำหนดขอบซ้ายของกระดาษ โดยส่งค่า 27, 108, n ซึ่ง n จะเป็นจำนวนคอลัมน์ที่จะเป็นขอบซ้ายของกระดาษ

9. Reset top - of form คือจะกำหนดความยาวของหน้ากระดาษเป็นจำนวนบรรทัด โดยส่งค่า 27, 67, n ซึ่ง n จะเป็นจำนวนบรรทัดใน 1 หน้า

สรุป

โปรแกรม SP.COM เป็นโปรแกรมที่ฝังตัวอยู่บนหน่วยความจำ จะอำนวยความสะดวกในการส่งคอนโทรลโค้ดเพื่อควบคุมเครื่องพิมพ์ ไม่ว่าคุณจะทำงานอะไรอยู่ เพียงแต่คุณกดคีย์ Ctrl + Right shift หลังจากที่โหลดโปรแกรมแล้ว SP.COM ก็พร้อมจะรับคำสั่งจากคุณ

รูปที่ 1 โปรแกรม SP.ASM ใช้ในงานควบคุมเครื่องพิมพ์

```
cr      equ     0dh
lf      equ     0Ah
;
code            segment para public 'code'
        assume cs:code
        org     100h
begin: jmp      initialize      ;goto initialization routine
;
banner          db 'The Set up Printer Utility, Copyright(C) 1989,'
                db 'SI''99 NO:11'
                db cr,lf,'$'
column_count    dw ?            ;width of window in columns
cursor_type     dw ?            ;cursor scan line definition
setup_status    db 0            ;indicates window is already active
display_mode    dw ?            ;current crt display mode
page_no         dw ?            ;current display page
attribute1      db 4Fh          ;window attribute
attribute2      db 70h
display_table db 20h,29h        ;display re-enable value for mode 2,3
video           dw 0B800h,0B900h,0BA00h,0BB00h ;staring address of video
                                               ;memory for CGA page 0-3
mono_video      dw 0B000h       ;video segment address for monochrome
old_kb_int      label dword
old_kb_addr     dw 2 dup (?)    ;storage for old keyboard interrupt
;
window_buffer label word
buffer_text     db 201,26 dup (205),187
                db 186,'    Printer Setup Menu    ',186
                db 186,'  Epson RX/FX/LX Printer  ',186
                db 199,26 dup (196),182
                db 186,' F1   Compressed Mode     ',186
                db 186,' F2   Expanded Mode       ',186
                db 186,' F3   Emphasized Mode     ',186
                db 186,' F4   Double-Strike Mode  ',186
                db 186,' F5   Elite Mode          ',186
                db 186,' F6   Miniature Mode      ',186
                db 186,' F7   Skip Perforation    ',186
                db 186,' F8   Indent Left Margin  ',186
                db 186,' F9   Reset Top-of-Form   ',186
                db 186,' F10  Reset Print Mode    ',186
                db 186,' Esc  Exit                ',186
                db 199,26 dup (196),182
                db 186,' Unshifted:   Toggle ON   ',186
                db 186,' Shifted:     Toggle OFF  ',186
                db 200,26 dup (205),188
                db 532 dup (?)
;
;Storage area for the text and attribute bytes that form
;the window image
;
window_byte     label byte
window_text     dw 532 dup (?)
;
;Control code string for all of the printer setup option
;
code_table      db 15,255,14 dup (0)        ;compressed mode on
                db 27,87,1,255,12 dup (0) ;expaned mode on
                db 27,69,255,13 dup (0)   ;emphasized mode on
                db 27,71,255,13 dup (0)   ;double-strike mode on
                db 27,77,255,13 dup (0)   ;elite mode on
                db 15,27,83,0,27,65,6,255,8 dup (0) ;miniature mode on
                db 27,78,12,255,12 dup (0)  ;perfrom skip on
```

รูปที่ 1 (ต่อ)

```
            db 27,108,10,255,12 dup (0) ;indent left margin
            db 27,67,66,255,12 dup (0)  ;reset top-of-form
            db 18,27,87,0,27,70,27,72   ;reset print mode
            db 27,80,27,84,27,50,255,0
;
            db 18,255,14 dup (0)        ;compressed off
            db 27,87,0,255,12 dup (0)   ;expaned mode off
            db 27,70,255,13 dup (0)     ;emphasized mode off
            db 27,72,255,13 dup (0)     ;double-strike mode off
            db 27,80,255,13 dup (0)     ;elite mode on
            db 18,27,84,27,50,255,10 dup (0) ;miniature mode off
            db 27,79,255,13 dup (0)     ;perfrom skip off
            db 27,108,0,255,12 dup (0) ;indent left margin
;
;-----------------------------------------------------------------
;The main body of the program when an interrupt 9 is generated
;from a keyboard. The main will check a keypress and compare
;to Ctrl+Right Shift.
;-----------------------------------------------------------------
new9h       proc   near
;
            push   ax
            push   bx
            push   cx
            push   dx
            push   si
            push   di
            push   ds
            push   es
            pushf              ;call to the old keyboard routine
            call   old_kb_int
            mov    ah,2        ;check status of shift key
            int    16h
            and    al,5        ;Ctrl and Rt shift depressed?
            cmp    al,5
            je     do_program  ;yes, then skip exit part
exit:       pop    es
            pop    ds
            pop    di
            pop    si
            pop    dx
            pop    cx
            pop    bx
            pop    ax
            iret
;
;Ctrl+Right Shift is pressed. A task is to check the status of window
;
do_program: push   cs          ;point DS and ES to code
            pop    ds
            push   cs
            pop    es
            cmp    setup_status,0 ;is the window already open?
            jne    exit           ;yes, ignore request
;
;Check current video mode. If video mode is 2,3,7 then set up
;all parameters
;else ignore request and exit
;
            mov    ah,15          ;get page and mode
            int    10h
            cmp    al,2
            je     prog0
            cmp    al,3
            je     prog0
            cmp    al,7
            je     prog0
            jmp    exit           ;not an acceptable mode, exit
prog0:      mov    setup_status,1 ;set flag to indicate window active
            xor    ah,ah          ;save mode number
            mov    display_mode,ax
            push   bx
            mov    bl,bh          ;save page number
            xor    bh,bh
            mov    page_no,bx
            pop    bx
            mov    ah,3           ;get cursor type
            int    10h
            mov    cursor_type,cx ;save it
            mov    ah,1           ;hide the cursor
            mov    ch,20h
            int    10h
;
;Now open the window by saving the content of video memory
;eneath the window
;and writing the window text to memory
;
            mov    bx,page_no  ;use BX as index into video
                               ;segment address table
            cmp    display_mode,7 ;is monochorme adapter?
            jne    prog1
            mov    bx,4           ;yes.
prog1:      shl    bx,1           ;BX = BX*2
            mov    ax,video[bx]   ;read segment from table
            mov    ds,ax          ;point DS to video memory
            cmp    display_mode,7 ;if monochorme then skip disable
            je     prog2
            call   video_disable  ;turn display off for
                                  ;snow-free writing
prog2:      lea    di,window_buffer ;point DI to buffer area to
                                    ;save screen
            mov    ch,28  ;define window dimention and location
            mov    cl,19
            mov    dh,2
```

รูปที่ 1 (ต่อ)

```
            mov   dl,26
            call  video2mem  ;then save screen to buffer
            push  ds         ;point ES to video memory
            pop   es
            push  cs         ;point DS to code
            pop   ds
            lea   si,window_text ;point to SI to window image
prog3:      mov   ch,28    ;define window dimention and location
            mov   cl,19
            mov   dh,2
            mov   dl,26
            call  mem2video    ;and the window to the screen
            cmp   display_mode,7 ;if monochorme then skip enable
            je    getkey
            call  video_enable   ;re-enable the video display
;
;Now window is written on the screen, wait for user command
;
getkey:     mov   ah,0         ;get a keypress
            int   16h
            or    al,al        ;is it an extended code?
            jz    extended_code  ;yes.
            cmp   al,27        ;ESC key?
            jne   getkey1      ;no, then beep
;
;ESC key is now pressed, so control is passed back
;to the application program.
;
            cmp   display_mode,7   ;is monochorme?
            je    prog4
            call  video_disable    ;turn off the display
prog4:      lea   si,window_buffer ;point SI to buffer area
            mov   ch,28
            mov   cl,19
            mov   dh,2
            mov   dl,26
            call  mem2video    ;and write the window to the screen
            cmp   display_mode,7   ;if monochorme then skip enable
            je    prog5
            call  video_enable     ;turn on the display
prog5:      mov   ah,1             ;restore cursor
            mov   cx,cursor_type
            int   10h
            mov   setup_status,0   ;reset window status
            jmp   exit             ;exit
;
;Illegal user command will beep and get a keypress again
;
getkey1:    call  beep
            jmp   getkey
;
;Interpret an extended code for going to the appropiate routine
;
extended_code:
            cmp   ah,59            ;less than F1?
            jb    getkey1          ;yes.
            cmp   ah,91            ;greater than Shift-F8?
            ja    getkey1          ;yes.
            cmp   ah,68            ;between F1 - F10?
            jbe   unshifted        ;yes.
            cmp   ah,84            ;between Shift-F1 - Shift-F9?
            jae   shifted          ;yes.
            jmp   getkey1          ;illegal command
;
;Scan code will be translated to the starting address
;of the control code
;
shifted:    sub   ah,15   ;adjust for shifted function key
unshifted:  sub   ah,59   ;adjust for unshifted function key
            mov   al,ah   ;convert index to word AX
            xor   ah,ah
            mov   cl,4
            shl   ax,cl   ;AX = AX*16
            add   ax,offset code_table ;and convert to offset address
            mov   si,ax
            call  lptistat   ;check for printer ready
            jc    getkey1    ;printer not ready
            mov   bl,255     ;use 255 for delimiter
            call  lprintz    ;send control code to printer
            jmp   getkey
;
new9h       endp
;
;-------------------------------------------------------------------
;VIDEO_ENABLE and VIDEO_DISABLE routines manipulate bit 3 of port 3D8h,
;the CGA mode control register, to tunr the display on or off.
;-------------------------------------------------------------------
video_disable proc  near
            mov   dx,03DAh   ;CGA status port
disable1:   in    al,dx      ;wait for vertical retrace
            test  al,8       ;is bit 3 set?
            jz    disable1   ;no, wait it
            mov   dx,03D8h   ;now disable the display
            mov   al,25h     ;by clearing bit 3 of the mode
            out   dx,al      ;control register
            ret
video_disable endp
;
video_enable  proc  near
            mov   dx,03D8h   ;CGA status port
            mov   bx,display_mode ;get value to re-enable display
            sub   bx,2
            mov   al,display_table[bx]
            out   dx,al      ;control register
```

รูปที่ 1 (ต่อ)

```
            ret
video_enable  endp
;
;-------------------------------------------------------------------
;VIDEO2MEM routine tranfers the contents of a part of video memory to
;memory buffer for storage.
;Entry:      DS    - video segment
;            ES:DI - memory buffer
;            DH,DL - row, column of upper left corner
;            CH    - width of window in columns
;            CL    - number of lines in window
;-------------------------------------------------------------------
video2mem     proc   near
              mov    al,ch
              xor    ah,ah
              mov    column_count,ax
              xor    ch,ch       ;CX = number of lines
              push   di
              call   video_offset  ;get a first cell address
              mov    si,di
              pop    di
v2m1:         push   si          ;save SI for next line
              push   cx          ;save line count
              mov    cx,colum_count  ;set CX for WRITELN
              call   writeln     ;tranfer one line
              pop    cx
              pop    si
              add    si,160      ;set SI for next line address
              loop   v2m1
              ret
video2mem     endp
;
;-------------------------------------------------------------------
;MEM2VIDEO writes a selected area fo memroy to the video display.
;Entry:      ES    - video segment
;            DS:SI - memory buffer
;            DH,DL - row, column of upper left corner
;            CH    - width of window in columns
;            CL    - number of lines in window
;-------------------------------------------------------------------
mem2video     proc   near
              mov    al,ch
              xor    ah,ah
              mov    column_count,ax
              xor    ch,ch          ;CX = number of lines
              call   video_offset   ;get a first cell address
m2v1:         push   di             ;save video starting address
              push   cx             ;save line count
              mov    cx,column_count  ;set CX for WRITELN
              call   writeln        ;tranfer one line
              pop    cx
              pop    di
              add    di,160         ;set DI for next line address
              loop   m2v1
              ret
mem2video     endp
;
;-------------------------------------------------------------------
;VIDEO_OFFSET calculates the offset into video memory of a character call.
;Entry:      DH,DL - row, column of cell
;Exit:       DI    - offset address
;-------------------------------------------------------------------
video_offset  proc   near
              mov    al,160
              mul    dh             ;row*160
              shl    dl,1           ;column*2
              xor    dh,dh          ;byte to word
              add    ax,dx          ;row*160 + column*2
              mov    di,ax          ;return in DI
              ret
video_offset  endp
;
;-------------------------------------------------------------
;WRITELN copies a string of words from one memory to another.
;Entry:      DS:SI - source
;            ES:DI - destination
;            CX    - number of word
;-------------------------------------------------------------
writeln       proc   near
              cld
write1:       movsw                 ;move one word
              loop   write1
              ret
writeln       endp
;
;-------------------------------------------------------------
;LPRINTZ sends a string of control code to LPT1: thru int 17h.
;Entry:      DS:SI - string address
;            BL    - delimiter
;-------------------------------------------------------------
lprintz       proc   near
              cld
lpz1:         lodsb                 ;get one byte
              cmp    al,bl          ;is it a delimiter?
              je     lpz2           ;yes, then exit
              xor    dx,dx          ;select LPT1:
              xor    ah,ah          ;print one byte
              int    17h
              jmp    lpz1
lpz2:         ret
lprintz       endp
;
;-----------------------------------------------
;LPT1STAT checks the current status of printer LPT1:.
```

รูปที่ 1 (ต่อ)

```
;Exit:       Carry clear - no error
;            Carry set   - error
;-------------------------------------------------
lptistat    proc   near
            xor    dx,dx        ;LPT1:
            mov    ah,2         ;get status
            int    17h
            test   ah,8         ;test bit 3 I/O error indicator
            jz     stat1        ;if clear, no error
            stc                 ;error
            ret
stat1:      clc                 ;no error
            ret
lptistat    endp
;
;-------------------------------------------------
;BEEP uses the 8253 timer chip to emit s short sound
;thru the PC's speaker.
;-------------------------------------------------
beep        proc   near
            mov    al,086h   ;tell 8253 that frequency data is
            out    43h,al    ;coming
            xor    al,al     ;send frequency 776.8 Hz
            out    42h,al
            mov    al,6
            out    42h,al
            in     al,61h    ;turn speaker on
            or     al,3
            out    61h,al
            mov    cx,6000h  ;time delay for sound duration
beep1:      loop   beep1
            in     al,61h    ;turn speaker off
            and    al,0FCh
            out    61h,al
            ret
beep        endp
;
;-----------------------------------------------------------------
;MEM2MEM transfers a non-overlapping block of memory one byte at a time.
;Entry:      DS:SI - source
;            ES:DI - destination
;            CX    - number of bytes
;-----------------------------------------------------------------
mem2mem     proc   near
            cld
mem1:       movsb             ;move one byte
            loop   mem1
            ret
mem2mem     endp
;
;-----------------------------------------------------------------
;Initialization routine set up all parameters and make a resident part
```

```
;-----------------------------------------------------------------
initialize  proc   near
;
            assume ds:code,es:code
            mov    dx,offset banner
            mov    ah,9
            int    21h
            mov    ah,15
            int    10h        ;check the current video mode
            cmp    al,7       ;if monochorme adapter then change
            jne    init0      ;attribute
            mov    attribute1,70h
            mov    attribute2,07h
init0:      cld
            lea    si,buffer_text  ;point SI to table of text
            lea    di,window_byte  ;and DI to storage area
            mov    cx,112          ;first four lines
            mov    al,attribute1   ;combine text and attribute1
init1:      movsb                  ;text
            stosb                  ;attribute
            loop   init1
            mov    cx,11           ;do 11 lines
init2:      push   cx
            movsb
            stosb
            mov    cx,26
            mov    al,attribute2
init3:      movsb
            stosb
            loop   init3
            movsb
            mov    al,attribute1
            stosb
            pop    cx
            loop   init2
            mov    cx,112
init4:      movsb
            stosb
            loop   init4
            mov    ax,3509h          ;get interrupt vector
            int    21h
            mov    old_kb_addr,bx    ;save old vector
            mov    old_kb_addr[2],es
            mov    dx,offset new9h   ;install new vector
            mov    ah,25h            ;set interrupt vector
            int    21h
            mov    dx,offset initialize ;end of resident section
            int    27h
initialize  endp
;
code        ends
            end    begin
```